AN10922

Symmetric key diversifications

Rev. 2.0 — 8 February 2017
165320

Application note
COMPANY PUBLIC

Document information

Info	Content
Keywords	MIFARE Plus, MIFARE DESFire, MIFARE SAM AV2, Key diversification, CMAC, TDEA, AES.
Abstract	This Application note describes CMAC based symmetric key diversification algorithms supported by NXP's MIFARE SAM AV2.

Revision history

Rev	Date	Description
2.0	20170208	General update
1.3	20100317	Re-organization, addition of examples.
1.2	20100129	Addition of AES-192, 2TDEA, 3TDEA key diversification algorithms.
1.1	20090813	Editorial changes, no content change.
1.0	20081112	Preliminary version.

Contact information

For more information, please visit: http://www.nxp.com

1. Introduction

Key diversification is a process of deriving the keys from a master (base) key using some unique input. Each card is getting a different value for each key, so that if one key is broken somehow (maybe from the terminal); the vulnerability is limited to that key on that card rather than the whole system being affected.

The diversified keys are generated and given (stored) to the PICC at its personalization phase, so all cards get unique keys. In the validation process, the POS terminal gets the information to generate the unique key for that unique card which is presented. MIFARE SAM AV2 can be an optimum secure solution for this key diversification process. The master (base) key can be stored securely in the MIFARE SAM AV2 and can be used to generate or use only the diversified keys.

MIFARE SAM AV2 supports two types of key diversification:

- old method, based on classical encryption and backwards compatible with SAM AV1, and
- new method, based on CMAC calculation

In this document, only the key diversification based on CMAC calculation is discussed, as it is the recommended one and new to the MIFARE SAM product. AES (128 and 192-bit key length) and TDEA (2-key and 3-key TDES) keys can be diversified using this CMAC based key diversification method.

In this document the algorithms are explained in a way that, they can be implemented easily in the SW in the installations without SAM today, but tomorrow using SAM.

All keys in a card can be derived from one master key however it is also possible to use a different master key for one set of keys versus another set of keys.

1.1 Abbreviations

Table 1. Abbreviations

Abbreviations	Meaning
AES	Advanced Encryption Standard
AID	Application ID
CBC	Cipher Block Chaining
CMAC	Cipher based MAC
DES	Data Encryption Standard
DF	DESFire
IV	Init Vector
LSB	Lowest Significant Bit
MAC	Message Authentication Code
MSB	Most Significant Bit
PCD	Proximity Coupling Device (reader/ writer unit)
PICC	Proximity Integrated Circuit Card
POS	Point Of Sales
TDEA	Triple Data Encryption Algorithm
UID	Unique Identification number

1.2 Examples presented in this document

The following symbols have been used to mention the operations in the examples:

= Preparation of data by SAM, PICC or host.

Please note, that the numerical data are used solely as examples. They appear in the text, in order to clarify the commands and command data.

Any data, values, cryptograms are expressed as hex string format if not otherwise mentioned e.g. 0x563412 in hex string format represented as “123456”. Byte [0] = 0x12, Byte [1] = 0x34, Byte [2] = 0x56.

2. Key Diversification

2.1 Construction

For diversification, the recommended way by NXP is to use the CMAC construction of an amount of data using a master key. See [CMAC].

The pre-requisite is that there is enough input "diversification data" in order to make it a MAC. A MAC is used rather than encryption to make it a one way function.

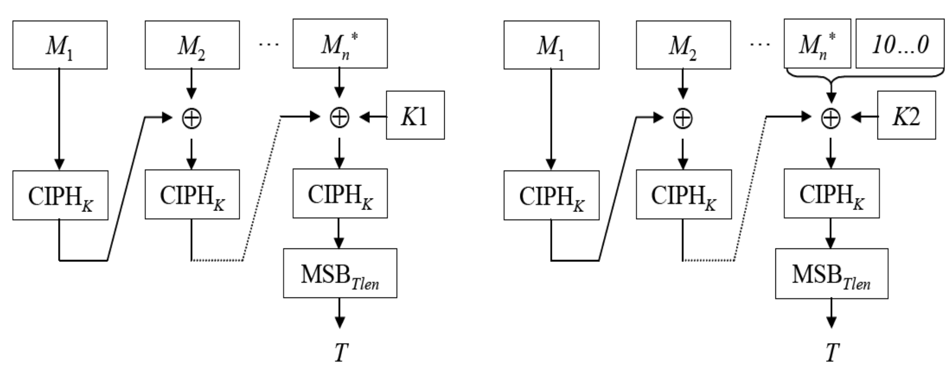

Fig 1. CMAC construction (2 cases: left without padding, right with padding)

Fig 1 illustrates the standard CMAC constructions (see [CMAC]) in two possible padding cases.

According to [CMAC], to avoid certain classes of attack (in the CMAC), the last block is modified before ciphering by being XORed with one of two possible "sub key" values (denoted K1 or K2), derived from an encryption of the zero vector under the key in use; the choice of which sub key to use is determined by whether the last message block contains padding or not.

These computations can be abstracted by the function ***CMAC (K, D, padded).*** In the context of the key derivations described further in this document another primitive is used because the padding is performed in a non-CMAC standard way. The corresponding computations can be abstracted by the function ***CMAC (K, D, Padded),*** where ***K*** is the key to be diversified, ***D*** the diversification input data and ***Padded*** is a Boolean flag that signals to the CMAC(.,.,.) function whether M had to be padded or not.

If the keys are to be diversified per card, it is recommended to use for the diversification input at least the UID of the card concatenated with e.g.

- For MIFARE Plus: the block number where the key is stored. Note however that if multi-sector authentication is desired, all keys that need to be the same need to be generated using same block number.
- For MIFARE DESFire: key number concatenated with application number.

Note: In this implementation, always two blocks (two times 16-byte for AES and two times 8-byte for TDEA) of message have been used.

2.2 AES-128 key

Input:

- 1 to 31 bytes of diversification input (let's name it "M")
- 16 bytes AES 128 bits master key (let's name it "K")

Output:

- 16 bytes AES 128 bits diversified key.

Algorithm:

1. Calculate CMAC input D:
2. D ← 0x01 || M || Padding
3. Padding is chosen such that D always has a length of 32 bytes. Padding bytes are according to the CMAC padding, i.e. 80h followed by 00h bytes. So the length of Padding is 0 to 30 bytes.
4. Calculate the Boolean flag 'Padded', which is true if M is less than 31 bytes long, false otherwise. The Boolean argument "Padded" is needed because it must be known in AES128CMAC which K1 or K2 is to be used in the last computation round.
5. Calculate output:
6. Diversified Key ← AES128CMAC (K, D, Padded)

Processing load:

One AES 128 key load, 3 AES 128 computations

Fig 2 shows the algorithm as a block diagram.

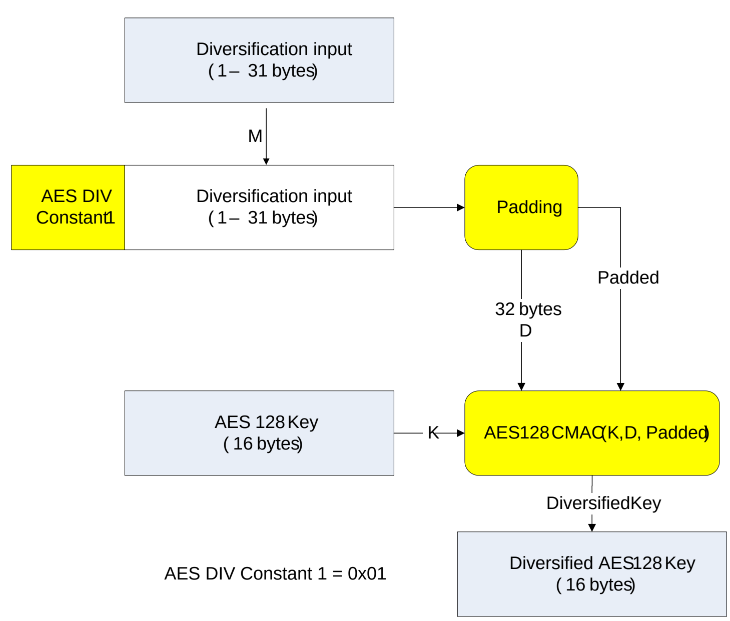

Fig 2. Diversification of 128-bit AES key

2.2.1 AES-128 key diversification example

Master key (K) = 00112233445566778899AABBCCDDEEFF, which will be diversified.

Table 2. Example – AES 128 key diversification

step	Indication		Data/ Message	Comment
CMAC sub key generation				
1	Master key (K)	=	001122334455667788 99AABBCCDDEEFF	The key, which is going to be diversified
2	K0	=	FDE4FBAE4A09E020 EFF722969F83832B	CIPHK(0b), AES (K, 16-byte 0s).
3	K1	=	FBC9F75C9413C041D FEE452D3F0706D1	The first sub key, see in [CMAC].
4	K2	=	F793EEB928278083B FDC8A5A7E0E0D25	The second sub key, see in [CMAC].
Diversified key generation				
5	UID	=	04782E21801D80	7-byte UID of PICC
6	Application ID	=	3042F5	3- byte DESFire AID
7	System Identifier	=	4E585020416275	ASCII of system identifier name
8	Diversification input (M)	=	04782E21801D803042 F54E585020416275	Data from step 5 to step 7. It doesn't matter how you make your diversification input, diversification input must be unique for unique PICC e.g. here the UID is unique and the same diversification input must be used in personalization and validation of the PICC. Maximum length of M is 31 bytes.
9	Add the Div Constant 1 at the beginning of M	=	0104782E21801D8030 42F54E585020416275	Div constant is fixed, must be 0x01 for AES 128 keys.
10	Do I need Padding	=	Yes	The algorithm always needs 32-byte block for AES; so far we have 18 bytes (step 9).
11	Padding	=	800000000000000000 0000000000	14-byte padding to make 32-byte block.
12	CMAC input D	=	0104782E21801D8030 42F54E585020416275 800000000000000000 0000000000	32 bytes
13	Last 16-byte is XORed with K2	=	0104782E21801D8030 42F54E5850204195E6 6EB928278083BFDC8 A5A7E0E0D25	As the padding is added the last block is XORed with K2, if padding is not added, then XORed with K1.
14	Encryption using K	=	351DB989A47CCA648 4CCE346FD5AE767A 8DD63A3B89D54B37 CA802473FDA9175	Standard AES encryption with IV = 00s in CBC mode
15	Diversified key	=	A8DD63A3B89D54B3 7CA802473FDA9175	Last 16-byte block. (CMAC)

If the length of M is more than 15 bytes, standard CMAC algorithm can be used, without taking care of padding, XOR-ing and encryption. The message for standard CMAC is then the data of step 9.

2.3 AES-192 key

Input:

- 1 to 31 bytes of diversification input (let's name it "M").
- 24 bytes AES 192 bits master key (let's name it "K").

Output:

- 24 bytes AES 192 bits diversified key.

Algorithm:

1. Calculate CMAC input D1 and D2:
2. D1 ← 0x11 || M || Padding
3. D2 ← 0x12 || M || Padding
4. Padding is chosen such that D1 and D2 always have a length of 32 bytes. Padding bytes are according to the CMAC padding, i.e. 80h followed by 00h bytes. So the length of Padding is 0 to 30 bytes.
5. Calculate the Boolean flag 'Padded', which is true if M is less than 31 bytes long, false otherwise. The Boolean argument "Padded" is needed because it must be known in AES192CMAC which K1 or K2 is to be used in the last computation round.
6. Calculate output:
7. DerivedKeyA ← AES192CMAC(K, D1, Padded)
8. DerivedKeyB ← AES192CMAC(K, D2, Padded)
9. DiversifiedKey ← first 8 bytes of DerivedKeyA || (next 8 bytes of DerivedKeyA XOR first 8 bytes of DerivedKeyB) || next 8 bytes of DerivedKeyB

Processing load:

One AES 192 key load, 6 AES 192 computations

If the special CMAC keys K1 and/or K2 can be reused from one to the following AES_CMAC operation, then we will need only 5 AES computations. But this depends on the HW implementation of the CMAC operation.

Fig 3 shows the algorithm as a block diagram.

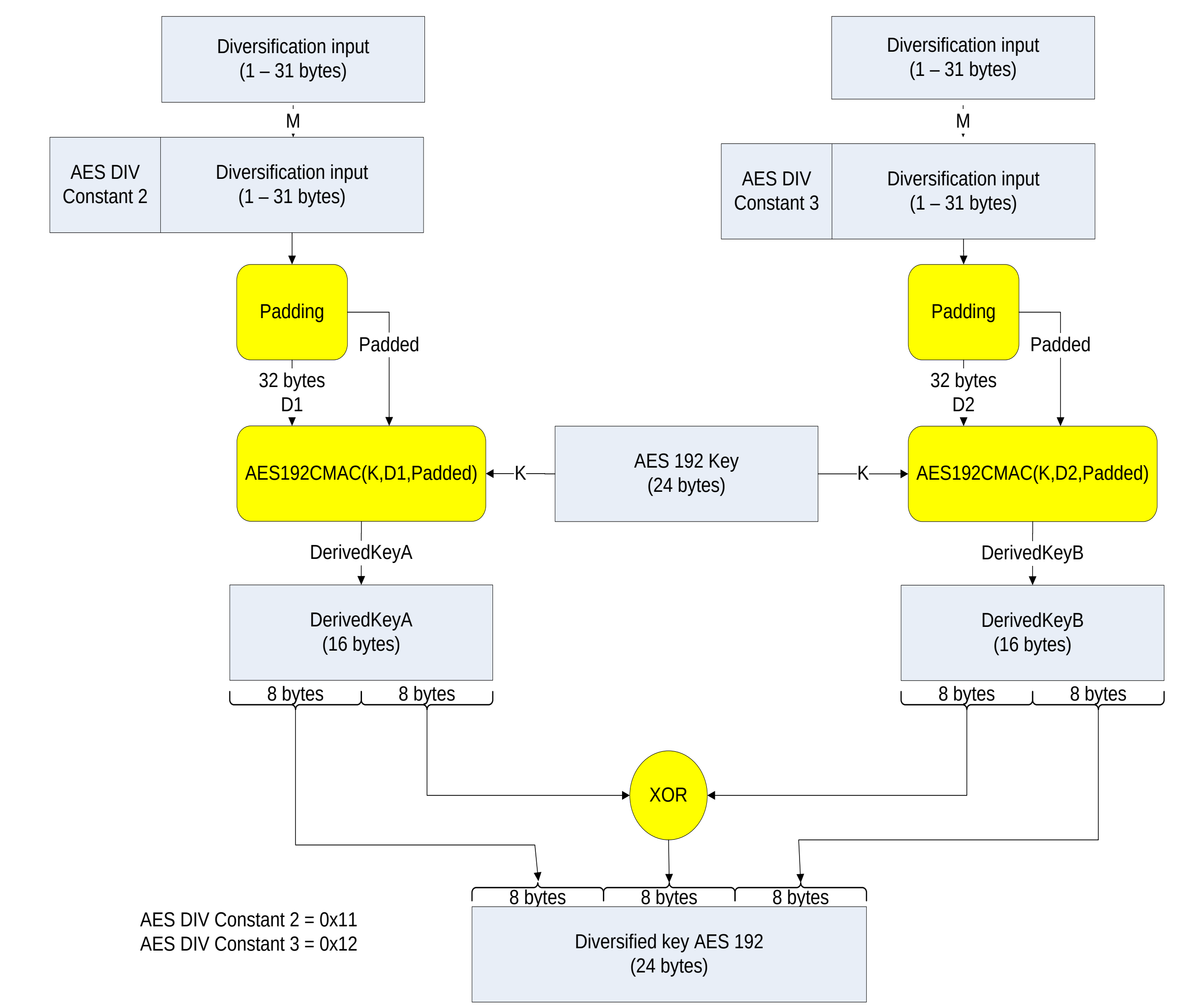

Fig 3. Diversification of 192-bit AES key

2.3.1 AES-192 key diversification example

Master key (K) = 00112233445566778899AABBCCDDEEFF0102030405060708, which will be diversified.

Table 3. Example – AES 192 key diversification

step	Indication		Data/ Message	Comment
CMAC sub key generation				
1	Master key (K)	=	0011223344556677889 9AABBCCDDEEFF01 02030405060708	The key, which is going to be diversified
2	K0	=	52DB5AFE7B64EFFA B1E92EEA983C5F73	CIPHK(0b), AES (K, 16-byte 0s).
3	K1	=	A5B6B5FCF6C9DFF5	The first sub key, see in [CMAC].

step	Indication		Data/ Message	Comment
			63D25DD53078BEE6	
4	K2	=	4B6D6BF9ED93BFEA C7A4BBAA60F17D4B	The second sub key, see in [CMAC].
Diversified key generation				
5	UID	=	04782E21801D80	7-byte UID of PICC
6	Application ID	=	3042F5	3- byte DESFire AID
7	System Identifier	=	4E585020416275	ASCII of system identifier name
8	Diversification input (M)	=	04782E21801D803042 F54E585020416275	Data from step 5 to step 7. It doesn't matter how you make your diversification input, diversification input must be unique for unique PICC e.g. here the UID is unique and the same diversification input must be used in personalization and validation of the PICC. Maximum length of M is 31 bytes.
9	Add the Div Constant 2 at the beginning of M	=	1104782E21801D8030 42F54E585020416275	Div constant 2 is fixed, must be 0x11 for AES 192 keys.
10	Do I need Padding	=	Yes	The algorithm always needs 32-byte block for AES; so far we have 18 bytes.
11	Padding	=	800000000000000000 0000000000	14-byte padding to make 32-byte block.
12	CMAC input D1	=	104782E21801D80304 2F54E5850204162758 000000000000000000 000000000	32 bytes
13	Last 16-byte is XORed with K2	=	1104782E21801D8030 42F54E585020412918 EBF9ED93BFEAC7A4 BBAA60F17D4B	As the padding is added the last block is XORed with K2, if padding is not added, then XORed with K1.
14	Encryption using K	=	C09ADDAE085769A6 E25DE29E51DA3669 CE39C8E1CD82D9A7 869FE6A2EF75725D	Standard AES encryption with IV = 00s in CBC mode
15	Diversified key A	=	CE39C8E1CD82D9A7 869FE6A2EF75725D	Last 16-byte block. (CMAC)
16	Add the Div Constant 3 at the beginning of M	=	1204782E21801D8030 42F54E585020416275	Div Constant 3 is fixed, must be 0x12 for AES 192 keys.
17	CMAC input D2	=	1204782E21801D8030 42F54E585020416275 800000000000000000 0000000000	Here the only difference is Div Constant 3, which is '12' fixed for AES 192.
18	Last 16-byte is XORed with K2	=	1204782E21801D8030 42F54E585020412918 EBF9ED93BFEAC7A4 BBAA60F17D4B	As the padding is added the last block is XORed with K2, if padding is not added, then XORed with K1.
19	Encryption using K	=	D052C22EA94BEFE1	Standard AES encryption with IV =

step	Indication		Data/ Message	Comment
			F748A9F5A675188A 38440F75A580E97E 176755EE7586E12C	00s in CBC mode
20	Derived key B	=	38440F75A580E97E 176755EE7586E12C	Last 16-byte block. (CMAC)
21	First 8-byte of derived key A	=	CE39C8E1CD82D9A7	
22	Last 8-byte of derived key A	=	869FE6A2EF75725D	
23	First 8-byte of derived key B	=	38440F75A580E97E	
24	Step 22 XOR step 23	=	BEDBE9D74AF59B23	
25	Last 8-byte of derived key B	=	176755EE7586E12C	
26	Diversified Key	=	CE39C8E1CD82D9A7 EDBE9D74AF59B2317 6755EE7586E12C	Step 21 + Step 24 + step 25

If the length of M is more than 15 bytes, standard CMAC algorithm can be used, without taking care of padding, X-ORing and encryption. The message for standard CMAC is then the data of step 9 and data of step 16.

2.4 2TDEA key

Input:

- 1 to 15 bytes of diversification input (let’s name it “M”)
- 16 bytes 2TDEA master key (let's name it “K”)

Output:

- 16 bytes 2TDEA diversified key.

Algorithm:

1. Calculate CMAC input D1 and D2:
2. D1 ← 0x21 || M || Padding
3. D2 ← 0x22 || M || Padding
4. Padding is chosen such that D1 and D2 always have a length of 16 bytes. Padding bytes are according to the CMAC padding, i.e. 80h followed by 00h bytes. So the length of Padding is 0 to 14 bytes.
5. Calculate the boolean flag ‘Padded’, which is true if M is less than 15 bytes long, false otherwise. The Boolean argument “Padded” is needed because it must be known in TDEACMAC which K1 or K2 is to be used in the last computation round.
6. Calculate output:

- DerivedKey1 = TDEACMAC(K, D1, Padded)
- DerivedKey2 = TDEACMAC(K, D2, Padded)
- 16-byte diversified key = DerivedKey1 || DerivedKey2.

Processing load: one 2TDEA key load, 6 2TDEA computations

We can reduce the TDEA operations to 5 if the CMAC K1 and/or K2 can be reused.

The Boolean argument “Padded” is needed because it must be known in TDEACMAC which K1 or K2 is to be used in the last computation round.

Remark: The master key can only be used about 1 million times if one wants to comply with SP 800-38B. This means that the construction suggested here can be used for 500000 cards. If more than 500000 cards are needed, and if duplicate keys are not acceptable for the application, a two level key diversification mechanism could be used.

Fig 4 shows the algorithm as a block diagram.

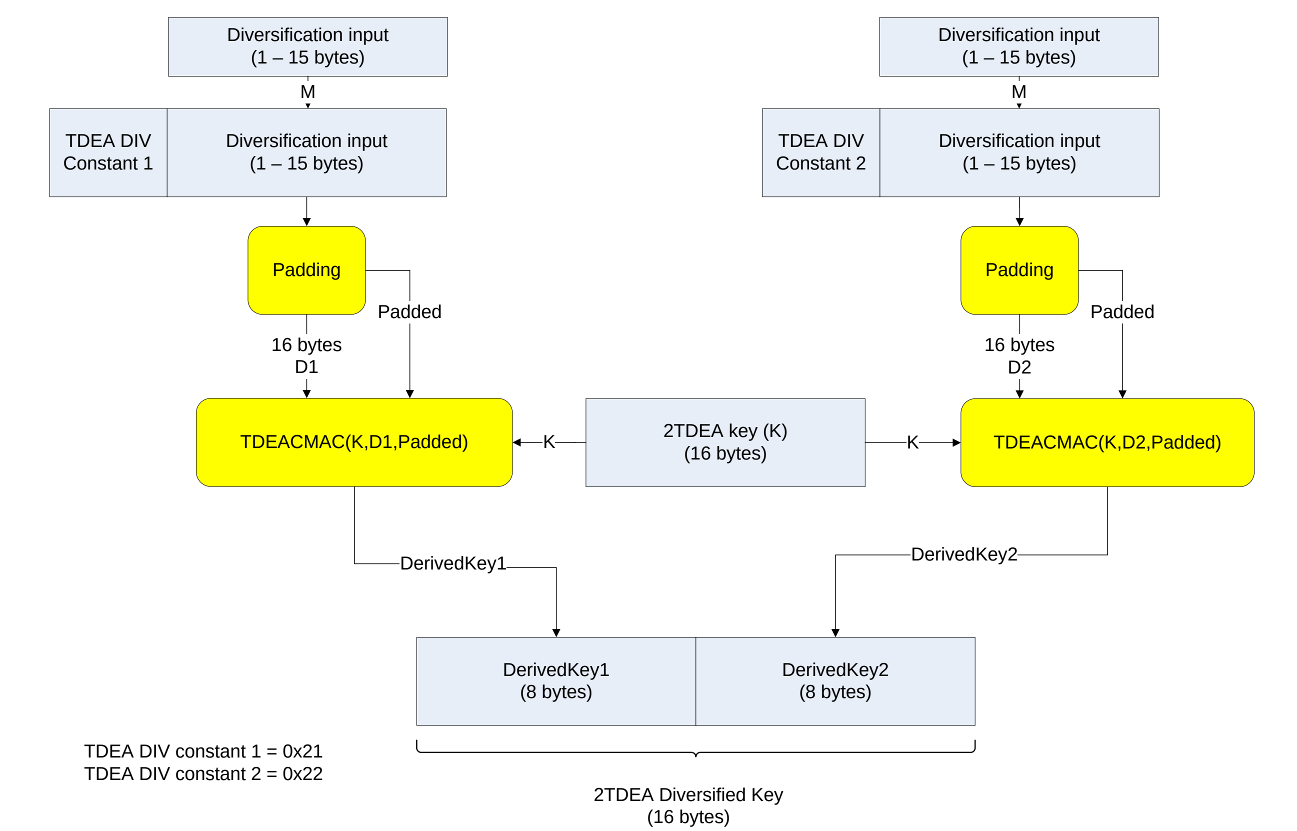

Fig 4. Diversification of 2TDEA key

MIFARE DESFire products store key version information in the lowest significant bits of the first 8 bytes 2TDEA key. If this versioning information is to be preserved, it is to be copied from the master key into the diversified key.

2.4.1 2TDEA key diversification example

Master key (K) = 00112233445566778899AABBCCDDEEFF, which will be diversified.

Table 4. Example – 2TDEA key diversification

step	Indication		Data/ Message	Comment
CMAC sub key generation				
1	Master key (K)	=	001122334455667788 99AABBCCDDEEFF	The key, which is going to be diversified
2	K0	=	FB09759972301AF4	CIPHK(0b), 2DEA (K, 8-byte 0s).
3	K1	=	F612EB32E46035F3	The first sub key, see in [CMAC].
4	K2	=	EC25D665C8C06BFD	The second sub key, see in [CMAC].
Diversified key generation				
5	UID	=	04782E21801D80	7-byte UID of PICC
6	Application ID	=	3042F5	3- byte DESFire AID
7	System Identifier	=	4E58502041	ASCII of system identifier name
8	Diversification input (M)	=	04782E21801D803042 F54E58502041	Data from step 5 to step 7. It doesn't matter how you specify your diversification input, the main thing, Diversification input must be unique for unique PICC e.g. here the UID is unique and the same diversification input must be used in personalization and validation of the PICC. This has to be up to 16 bytes.
9	Add the TDEA Div Constant 1 at the beginning of M	=	2104782E21801D8030 42F54E58502041	It is fixed, must be '21' for 2TDEA keys.
10	Do I need Padding	=	No	The algorithm always needs 16-byte block for TDEA, Here message is 16 bytes.
11	CMAC input D1	=	2104782E21801D8030 42F54E58502041	16 bytes
12	Last 16-byte is XORed with K1	=	2104782E21801D80C 6501E7CBC3015B2	As the padding is NOT added the last block is XORed with K1, if padding is added, then XOR with K2.
13	Encryption using K	=	5B7B81DCDE98A6BE 16F8597C9E8910C8	Standard TDEA encryption with IV = 00s in CBC mode
14	Derived Key 1	=	16F8597C9E8910C8	CMAC
15	Add the TDEA Div Constant 2 at the beginning of M	=	2204782E21801D8030 42F54E58502041	
16	Do I need Padding	=	No	
17	CMAC input D1	=	2204782E21801D8030 42F54E58502041	16 bytes
18	Last 8-byte is XORed with K1	=	2204782E21801D80C 6501E7CBC3015B2	As the padding is NOT added the last block is XORed with K1, if padding is added, then XOR with K2.
19	Encryption using K	=	D2292CCE0B8106CE 6B9648D006107DD7	Standard TDEA encryption with IV = 00s in CBC mode

step	Indication		Data/ Message	Comment
20	Derived Key 2	=	6B9648D006107DD7	CMAC
21	2TDEA diversified key (without restoring the key version)	=	16F8597C9E8910C86 B9648D006107DD7	Step 15 + step 20
The lowest significant bit of every key byte is not used in DES calculation. MIFARE DESFire and SAMs use the lowest significant bit of first eight bytes key as the key version. In this example the version of master key = 0x55 (01010101_b). These version bits are required to insert in the diversified key as well, to make the same key version for master key and diversified keys.				
22	2TDEA diversified key (after inserting the key version)	=	16F9587D9E8910C9 6B9648D006107DD7	

If the length of M is more than 7 bytes, standard CMAC algorithm can be used, without taking care of padding, XOR-ing and encryption. The message for standard CMAC is then the data of step 9 and data of step 15.

2.5 3TDEA key

Input:

- 1 to 15 bytes of diversification input (let's name it "M")
- 24 bytes 3TDEA master key (let's name it "K")

Output:

- 24 bytes 3TDEA diversified key.

Algorithm:

1. Calculate CMAC input D1, D2 and D3:
2. D1 ← 0x31 || M || Padding
3. D2 ← 0x32 || M || Padding
4. D3 ← 0x33 || M || Padding
5. Padding is chosen such that D1, D2 and D3 always have a length of 16 bytes. Padding bytes are according to the CMAC padding, i.e. 80h followed by 00h bytes. So the length of Padding is 0 to 14 bytes.
6. Calculate the Boolean flag 'Padded', which is true if M is less than 15 bytes long, false otherwise. The Boolean argument "Padded" is needed because it must be known in TDEACMAC which K1 or K2 is to be used in the last computation round.
7. Calculate output:

- DerivedKey1 = TDEACMAC(K, D1, Padded)
- DerivedKey2 = TDEACMAC(K, D2, Padded)
- DerivedKey3 = TDEACMAC(K, D3, Padded)
- 16-byte diversified key = DerivedKey1 || DerivedKey2 || DerivedKey3.

Processing load: one 3TDEA key load, 9 3TDEA computations

Remark: The master key can only be used about 1 million times if one wants to comply to SP 800-38B. This means that the construction suggested here can be used for about 330000 cards. If more than 330000 cards are needed, and if duplicate keys are not acceptable for the application, a two level key diversification mechanism is used.

The Boolean argument "Padded" is needed because it must be known in TDEACMAC which K1 or K2 is to be used in the last computation round.

Fig 5 shows the algorithm as a block diagram.

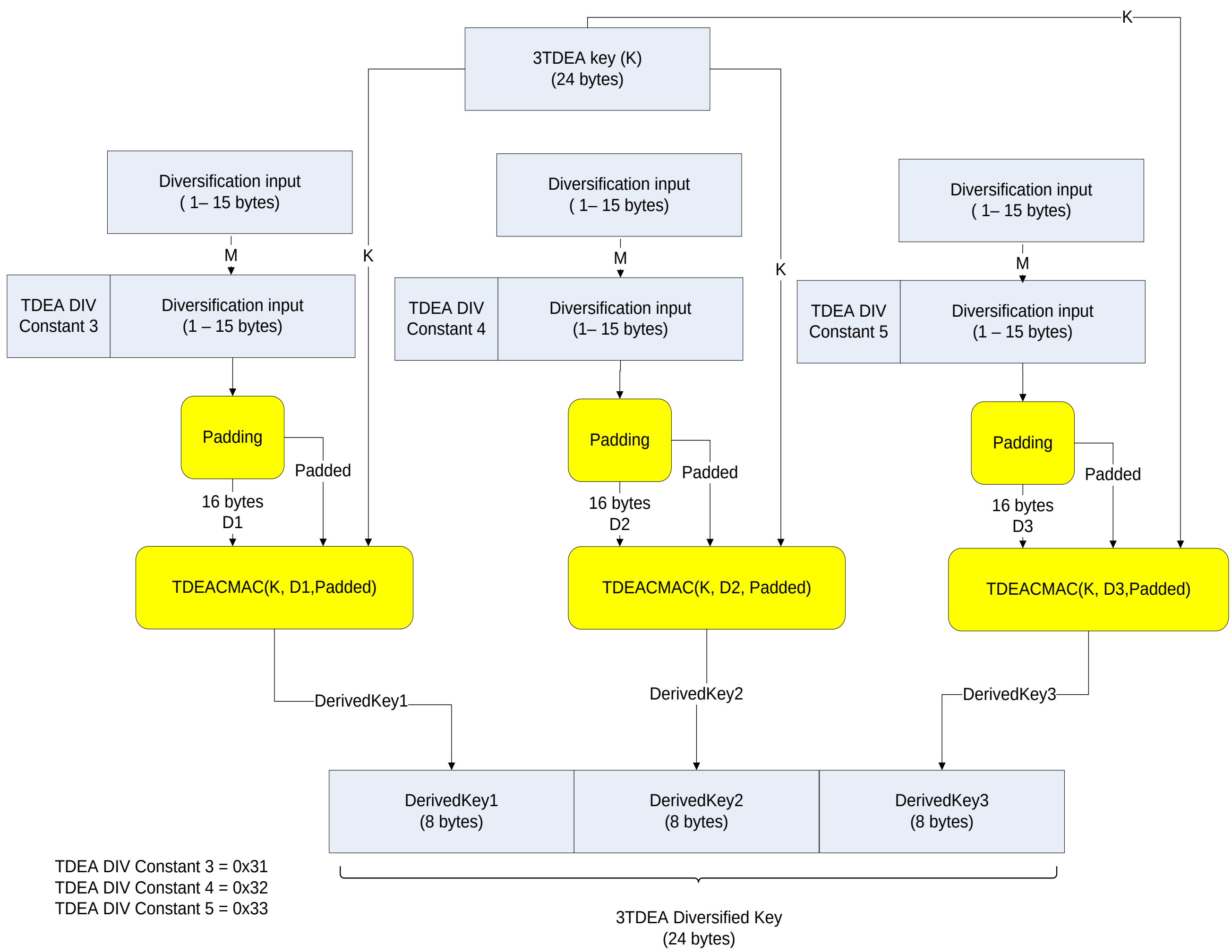

Fig 5. Diversification of 3TDEA key

MIFARE DESFire products store key version information in the lowest significant bits of the first 8 bytes 3TDEA key. If this versioning information is to be preserved, it is to be copied from the master key into the diversified key.

2.5.1 3TDEA key diversification example

Master key (K) = 00112233445566778899AABBCCDDEEFF0102030405060708, which will be diversified.

Table 5. Example – 3TDEA key diversification

step	Indication		Data/ Message	Comment
CMAC sub key generation				
1	Master key	=	00112233445566778899AABBCCDDEEFF010203040506070 8	The key, which is going to be diversified
2	K0	=	51F6AC7C734A0DE5	CIPHK(0b), 2DEA (K, 8-byte 0s).
3	K1	=	A3ED58F8E6941BCA	The first sub key, see in [CMAC].
4	K2	=	47DAB1F1CD28378F	The second sub key, see in [CMAC].
Diversified key generation				
5	UID	=	04782E21801D80	7-byte UID of PICC
6	Application ID	=	3042F5	3- byte DESFire AID
7	System Identifier	=	4E5850	ASCII of system identifier name
8	Diversification input (M)	=	04782E21801D803042F54E5850	Data from step 5 to step 7. It doesn't matter how you specify your diversification input, the main thing, Diversification input must be unique for unique PICC e.g. here the UID is unique and the same diversification input must be used in personalization and validation of the PICC. This has to be up to 16 bytes.
9	After inserting TDEA Div constant 3	=	3104782E21801D803042F54E5850	It is fixed, must be '31' for 3TDEA keys.
10	Do I need Padding	=	Yes	The algorithm always needs 16-byte block for TDEA, here message is 14 bytes.
11	CMAC input D1	=	3104782E21801D803042F54E58508000	8000 padding added
12	Last 8-byte is XORed with K2	=	3104782E21801D80779844BF9578B78F	As the padding is added the last block is XORed with K2, if padding is NOT added, then XOR with K1.
13	Encryption using K	=	4C294A83A6829EC12F0DD03675D3FB9A	Standard TDEA encryption with IV = 00s in CBC mode
14	Derived Key 1	=	2F0DD03675D3FB9A	CMAC
15	After inserting TDEA Div constant 4 in M	=	3204782E21801D803042F54E5850	It is fixed, must be '32' for 3TDEA keys.
16	Do I need Padding	=	Yes	The algorithm always needs 16-byte block for TDEA, here message is 14 bytes.
17	CMAC input D2	=	3204782E21801D803042F54E58508000	8000 padding added
18	Last 8-byte is	=	3204782E21801D8077	Diversification constant and

step	Indication		Data/ Message	Comment
	XORed with K2		9844BF9578B78F	diversification input. Here the constant must be '32'
19	Encryption using K	=	41A9459AB5B209905 705AB0BDA91CA0B	Standard TDEA encryption with IV = 00s in CBC mode
20	Derived Key 2	=	5705AB0BDA91CA0B	CMAC
21	After inserting TDEA Div constant 5 in M	=	3304782E21801D8030 42F54E5850	It is fixed, must be '33' for 3TDEA keys.
22	Do I need Padding	=	Yes	The algorithm always needs 16-byte block for TDEA, here message is 14 bytes
23	CMAC input D3	=	3304782E21801D8030 42F54E58508000	8000 padding added
24	Last 8-byte is XORed with K2	=	3304782E21801D8077 9844BF9578B78F	Diversification constant and diversification input. Here the constant must be '33'
25	Encryption using K	=	7FABF1B71419AF155 5B8E07FCDBF10EC	Standard TDEA encryption with IV = 00s in CBC mode
26	Derived Key 3	=	55B8E07FCDBF10EC	CMAC
27	Diversified 3TDEA key (without restoring the key version)	=	2F0DD03675D3FB9A5 705AB0BDA91CA0B5 5B8E07FCDBF10EC	24-byte 3TDEA key. (Step 14 + step 20 + step 26).
The lowest significant bit of every key byte is not used in DES calculation. MIFARE DESFire and SAMs use the lowest significant bit of first eight bytes key as the key version. In this example the version of master key = 0x55 (01010101_b). These version bits are required to insert in the diversified key as well, to make the same key version for master key and diversified keys.				
28	Diversified 3TDEA key (after restoring the key version)	=	2E0DD03774D3FA9B5 705AB0BDA91CA0B5 5B8E07FCDBF10EC	

If the length of M is more than 7 bytes, standard CMAC algorithm can be used, without taking care of padding, XOR-ing and encryption. The message for standard CMAC is then the data of step 9, step 15 and step 21.

3. Conclusion

The master keys must be stored securely if the algorithms are implemented in software. MIFARE SAM AV2 offers secure storage of the master keys and dynamic diversifications. For the optimum security, using MIFARE SAM AV2 can be the best solution. The user shall take care for defining his master keys, shall avoid the weak keys whenever necessary. Neither the SAM nor the algorithms analyze the keys. NXP recommends using AES instead of TDEA.

4. Legal information

4.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

4.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

4.3 Licenses

ICs with DPA Countermeasures functionality

LICENSED DPA COUNTERMEASURES™

NXP ICs containing functionality implementing countermeasures to Differential Power Analysis and Simple Power Analysis are produced and sold under applicable license from Cryptography Research, Inc.

4.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

MIFARE — is a trademark of NXP B.V.

MIFARE Plus — is a trademark of NXP B.V.

MIFARE DESFire — is a trademark of NXP B.V.

5. List of figures

6. List of tables

7. Contents

Date of release: 8 February 2017
165320
Document identifier: AN10922